Stakeholders' information needs for artifacts and their dependencies in a real world context

Müller, Sebastian C ; Fritz, Thomas

Abstract: In the evolution of software, stakeholders continuously seek and consult various information artifacts and their interdependencies to successfully complete their daily activities. While a lot of research has focused on supporting stakeholders in satisfying various information needs, there is little empirical evidence on how these information needs manifest themselves in the context of professional software development teams of real world companies. To investigate the information needs of the different stakeholder roles involved in software evolution activities, we conducted an empirical study with 23 participants from two professional development teams of one company. The analysis of the gathered data shows that information needs exhibit a crosscutting nature with respect to stakeholder role, activity, artifacts and even fragments of artifacts. We also found that the dependencies between information artifacts are important for the successful performance of software evolution activities, but often not captured explicitly. The lack of an explicit representation of these interdependencies often result in difficulties identifying dependent artifacts and additional communication effort. Based on our findings, we suggest ways to better support stakeholders with their information needs.

DOI: https://doi.org/10.1109/ICSM.2013.40

Stakeholders' Information Needs for Artifacts and their Dependencies in a Real World Context

Sebastian C. Müller
Department of Informatics
University of Zurich, Switzerland
smueller@ifi.uzh.ch

Thomas Fritz
Department of Informatics
University of Zurich, Switzerland
fritz@ifi.uzh.ch

***Abstract*—In the evolution of software, stakeholders continuously seek and consult various information artifacts and their interdependencies to successfully complete their daily activities. While a lot of research has focused on supporting stakeholders in satisfying various information needs, there is little empirical evidence on how these information needs manifest themselves in the context of professional software development teams of real world companies. To investigate the information needs of the different stakeholder roles involved in software evolution activities, we conducted an empirical study with 23 participants from two professional development teams of one company. The analysis of the gathered data shows that information needs exhibit a crosscutting nature with respect to stakeholder role, activity, artifacts and even fragments of artifacts. We also found that the dependencies between information artifacts are important for the successful performance of software evolution activities, but often not captured explicitly. The lack of an explicit representation of these interdependencies often result in difficulties identifying dependent artifacts and additional communication effort. Based on our findings, we suggest ways to better support stakeholders with their information needs.**

***Index Terms*—information needs, crosscutting nature, stakeholder role, activity, artifact, fragment, missing link**

I. Introduction

Software evolution requires the coordination and collaboration of multiple stakeholders performing a variety of activities (*e.g.*, [1], [2]), where many of these activities deal with already existing code, such as analyzing and fixing bugs in a system. In this process, large amounts of information are continuously evolved and recorded in various kinds of artifacts, such as source code, bug reports or requirements documents. These information artifacts and their dependencies are then continuously consulted by the stakeholders of the system, to successfully complete their activities. For instance, a software developer might have to understand a bug reported by a software tester and find the right part in the source code to fix the bug, or a software tester might have to examine the requirements specified by the requirements engineer to see if his newly created test cases actually cover them.

Although there is a lot of research to support stakeholders with their information needs, even across multiple artifacts (*e.g.*, [3], [4]) or to recover the often implicit dependencies between artifacts (*e.g.*, [5], [6]), there is little evidence on how these information needs manifest themselves in the workspace of real world companies. Existing evidence on information needs mainly stems from retrospective analyses of repositories (*e.g.*, [7], [8], [9]) and studies predominantly conducted in a lab setting that either focus on a single kind of information artifact, a single activity or a single stakeholder role (*e.g.*, [10], [11], [12], [13], [14], [15]).

To investigate stakeholders' information needs and their manifestation in a real world context, we conducted an empirical study with 23 participants from two different software development teams of one European company[1]. This study was composed of two parts, a diary study and a follow-up interview. In this study, we focused on the following two research questions:

- What are the characteristics of information artifacts needed by the different kinds of stakeholder roles involved in the daily software evolution activities?
- How are these artifacts interdependent?

We found that information needs exhibit a crosscutting nature with respect to stakeholder role, activity, artifacts and even fragments of artifacts. Furthermore, we found that the dependencies between various information artifacts are important for stakeholders to successfully perform their activities but are often not explicitly represented, causing difficulties for individuals to understand and identify dependencies as well as additional communication effort.

This paper makes the following contributions:

- it identifies characteristics of the information artifacts needed by stakeholders in the context of a real world company;
- it identifies characteristics of the dependencies between these artifacts needed by stakeholders for their daily activities; and
- it provides a discussion and implications of the crosscutting nature of information needs.

II. Related Work

Work related to our approach can broadly be categorized into three major areas: first, studies investigating stakeholders' activities in the software evolution process as well as the information sharing and communication practice during these activities, second, research focusing on the information needs

[1]Due to intellectual property, we are not able to disclose more specific information about the company.

of stakeholders, and third, research looking at the dependencies between information artifacts that are relevant for the activities of a stakeholder. In the following, we will provide an overview over each of these areas by sampling the work in each area.

Various studies have looked at stakeholder activities in the software evolution process. For instance, Perry *et al.* [1] conducted two studies to investigate how software developers spend their time and found that software developers spend more than half of their time on non-coding activities. LaToza *et al.* [16] provide an overview on the typical activities of software developers as well as the practices and tools they use to perform these activities based on a set of surveys and interviews. Singer *et al.* [17] performed four different studies to investigate the daily activities of software developers and found that they spend a lot of time on reading or writing documentation, interacting with the source code and various search queries. Schröter *et al.* [18] investigated the communication between software developers in a software project, focusing on the factors that influence the communication behavior around change sets. Similarly, Aranda *et al.* [8] investigated the coordination needs and patterns of developers by looking at the bug reports of resolved bug fixes. Finally, De Souza *et al.* [19] conducted a field study to investigate the interdependencies in the process of software development, in particular, assessing the approaches a software development team uses to coordinate their work flow. Different to our work, all these studies do not examine the characteristics of the information needed to perform an activity and are often limited to the software developers or a single activity rather than looking at the multiple stakeholders involved in software evolution.

Other research has focused on the information needs and the questions of individual stakeholder roles, in particular software developers[2]. This research area can be further divided into studies conducted with software developers and retrospective analysis of repositories. In the studies with software developers, Sillito *et al.* [10], for instance, observed software developers while performing change tasks and identified a catalogue of common source code related questions that developers ask themselves. LaToza *et al.* [12] investigated more specifically the reachability questions that developers ask for the code and that are difficult to answer. Ko *et al.* [11] identified 21 more general types of questions that software developers ask themselves on a daily basis, the information developers seek to answer these questions and the difficulties developers have to acquire them. Fritz *et al.* [13] conducted a study to specifically identify the questions that require multiple kinds of information artifacts and are often difficult or infeasible to answer. To investigate a stakeholder's information needs using a repository, Breu *et al.* [9] looked at bug reports and investigated the information needs of software developers documented in these bug reports. They identified eight question categories from the questions asked in bug reports and found that the information needs evolve with the bug life cycle. Erdem *et al.* [7] examined messages posted to the Usenet newsgroup to analyze what information people ask for and came up with a question classification scheme. In a more recent study, Treude *et al.* [20] looked at Stack Overflow, analyzed the tags that are used there for questions and found ten categories of questions, such as error and how-to questions. All of these studies mostly paid little to no attention on how the information needs manifest themselves in the team context of real world companies and either focused on a single stakeholder role or a single activity. Studies that examined professional developers were conducted by Roehm *et al.* [14] and Seaman [15]. In their study, Roehm *et al.* [14] investigated the strategies that developers follow to comprehend software as well as the information artifacts and tools developers use in the comprehension process. Seaman [15] paid more attention on software maintainers' information needs and conducted a survey to assess how software maintainers acquire the information they need to perform their maintenance tasks. Different to these two studies, we looked at multiple stakeholder roles.

Dependencies between information artifacts is a widely discussed topic, particularly concerning the recovery of missing dependencies between various information artifacts for traceability reasons. There are a variety of approaches using information retrieval methods to recover the links between multiple information artifacts, such as documentation and source code (*e.g.* [5], [21], [22]). In most of these papers, the authors assume that stakeholders consider these interdependencies as important, since they support stakeholders while performing a number of typical software evolution and maintenance tasks. However, to the best of our knowledge, we do not know of any research that investigates these interdependencies and their manifestations in a real world context.

III. Empirical Study Design

The goal of this study is to investigate the information needs of the multiple stakeholders, such as software developers and requirements engineers, in the context of a real world company. In particular, we were interested in the following two question:

- What are the characteristics of information artifacts needed by the different kinds of stakeholder roles involved in the daily software evolution activities?
- How are these artifacts interdependent?

To study these questions in the context of evolving software, we conducted an empirical study with 23 participants from two professional software development teams of a big European company[3]. We chose these two teams due to the access we were granted to all stakeholders. The study was composed of a diary study to gather mainly quantitative data and a follow-up interview for more qualitative data. During the follow-up interviews we asked questions to get more detailed insights

[2]With the term software developer we will refer to the stakeholder role that is focused on the coding aspect, often also called a programmer.

[3]Due to intellectual property, we are not able to disclose more specific information about the company.

about the data gathered during the diary study. We decided to conduct a diary study and against performing another kind of study, such as observations, since we wanted to find out more about stakeholders' typical software evolution activities without being too intrusive.

A. Team Structure and Subjects

All study participants were part of two software development teams of a big engineering company. The company employs tens of thousands of people all over the world, not all in software development. Each of the two teams consisted of one line manager, responsible for a product line and overseeing several projects, two project managers, responsible for a single project and his project team, one to two requirements engineers, six to seven software developers and two to four software testers. These roles are a subset of the fairly extensive stakeholder roles identified by Acuna *et al.* [23] and Yilmaz *et al.* [24]. The size of the teams, 13 and 16, is comparable to other software development teams in industry as reported by [25] and [26]. Except for one of the line managers and some software testers, all members of each team shared an office space.

Initially, we contacted all 29 members from both teams. To get an unbiased sample of stakeholders, we did not preselect any of the team members. 23 of the 29 team members were willing to participate in our study. The other six team members did not participate, mainly due to lack of available time. The 23 participants included two project managers, two line managers, three requirements engineers, five software testers, and eleven software developers. Given the original role distribution on the teams, these participants presented a representative sample of teams and stakeholder roles on the teams involved in the software development process at this company, and also a representative sample of stakeholder roles involved in software development teams in general. From the 23 participants in the diary study, three subjects did not have time to participate in the follow-up interview.

An overview of all participants is presented in Table I. Participants had an average of 12.2 years (ranging from 1.5 to 25 years) of experience in the software engineering domain and an average of 7.9 years (ranging from 1 to 21 years) of experience performing their specific stakeholder role.

B. Development Process

Both teams we observed during our study followed the same agile software development process that divides the development into release cycles typically lasting three months. A release cycle is further split into several iterations each lasting several weeks. During each iteration, new features, feature improvements as well as bug fixes are integrated into the software. At the end of each iteration, an internal beta release is created for internal testing purposes. Finally, each iteration has several milestones, each specifying a list of planned change requests to be completed for the milestone. These milestones are reviewed for quality, performance and completeness reasons.

TABLE I
OVERVIEW OF STUDY PARTICIPANTS (*Role* INDICATING THE PARTICIPANTS' AVERAGE EXPERIENCE IN THEIR ROLE; *SE* THEIR AVERAGE EXPERIENCE IN SOFTWARE DEVELOPMENT; * INDICATING THAT THE SUBJECT DID NOT PARTICIPATE IN THE FOLLOW-UP INTERVIEW)

Role	Subjects	Experience (in years)	
		Role	SE
Team 1			
Software developer (SD)	D1, D2, D5, D8, D10*	12.8	15.0
Software tester (ST)	T3	5.0	8.0
Requirements engineer (RE)	R1	8.0	15.0
Project manager (PM)	P2*	1.0	12.0
Line manager (LM)	M2	1.5	1.5
Team 2			
Software developer (SD)	D3, D4, D6, D7, D9, D11*	9.8	13.2
Software tester (ST)	T1, T2, T4, T5	4.9	10.8
Requirements engineer (RE)	R2, R3	5.5	16.5
Project manager (PM)	P1	10.0	15.0
Line manager (LM)	M1	2.0	7.0

All code changes during software development are based on change requests. A change request can be a bug report, a feature request or an improvement. These change requests are reported from all stakeholders that are involved in the development of the software project as well as people using the software in the field. At least once a week these requests are analyzed, prioritized and assigned to software developers to resolve them.

C. Study Methods

Our study was composed of a diary study with a follow-up interview. We asked the participants to complete an online survey, a diary, at the end of each day over a period of six workdays. The survey questions focused on participants' daily activities, the information they worked with to perform these activities and where they retrieved the information. The online survey contained a total of 10 questions and took participants an average of 14.22 minutes to complete. After completing the diary study, we scheduled and conducted an in-person interview with each participant, except for the three participants that were not available for the interview. The interview took an average of 39 minutes per participant and was designed to gather more detailed insights on a stakeholder's daily activities and the information he works with. We chose this combination of starting with an online diary study to get a general understanding of stakeholders' typical activities, information needs and the frequency they occur. In the second part, we used the answers from the diary study to guide the follow-up interviews for gathering more detailed information as well as answering open questions from and clarifying vague responses to the diary study[4].

Diary Study. For each day of the diary study, we asked the participants three sets of questions after asking them about their role. First, they were asked about the top five activities they spent the most time on during their work day, second,

[4]The survey used for the diary study and the questions guiding the semi-structured interview can be found at: http://www.ifi.uzh.ch/seal/people/mueller/info-needs

the participants were asked about the information sources, documents and tools they used for each one of the activities mentioned, and finally, we asked participants to state the stakeholders, in particular their roles, they interacted with during their day.

Interview Study. The follow-up interview was conducted as a semi-structured interview. This means, we prepared a set of questions for the interview, but did not strictly follow these questions. Instead, we used them as a general guidance to gain further insights into the answers the participants provided during the diary study. In particular, the questions focused on the participants' activities, their information needs and how the participants acquire, manage and share the needed information artifacts for the performed activities. The interview was recorded and notes were taken manually. Directly after each interview, the protocol was transcribed and augmented with additional comments by the interviewer.

D. Data Analysis

Over the course of the study, we collected a large amount of data consisting primarily of answers to the diary study and transcripts of the follow-up interviews. Over all participants, we collected 26 diary entries for software testers, 54 for software developers, 12 for requirements engineers, 8 for project managers and 8 for line managers. Despite providing the participants the flexibility to start the diary study when they had time over the period of a month, not all participants were able to fill in the diary study for the whole 6 work days due to availability. From these entries, we gathered detailed descriptions of 403 activities and the corresponding information artifacts used for these activities.

In order to analyze the collected data from the diary and the interview study, we used a grounded theory approach [27]. First, we followed an open coding approach to develop concepts and categories from the interview transcriptions. We then used axial coding to relate these concepts and categories to each other. Finally, we identified important categories that were brought up by most study participants. Finally, by using selective coding, we systematically related portions of our data to the identified categories. Our findings are presented in Section IV and are discussed in Section V.

E. Threats

There are several threats to the validity of our study, mainly to the external and conclusion validity.

External Validity. Since we gathered data from a single company, the generalizability of our findings might be limited. We tried to mitigate the risk by observing two different software development teams of that company. Also, since the two observed teams exhibit characteristics in terms of size and roles similar to others reported in the research literature, *e.g.* [24], [25], we believe that the impact of the single company is not a substantial limitation.

Second, we only investigated the typical activities during a period of six work days which, again, might limit the generalizability of our findings. However, during the follow-up interview, we asked participants and found out that most of the reported activities are typical and representative for their overall activities.

Conclusion Validity. We only used one interviewer to conduct the interviews and transcribe the interview protocol. Additionally, we also only used one coder to categorize our field notes. To minimize the risk of misinterpretation and misunderstanding, we audio recorded the interviews and reviewed our field notes, as well as the categorization with a second investigator.

IV. Results

Based on the analysis of our qualitative and quantitative data, we identified several observations on stakeholders' activities and on the characteristics of stakeholders' information needs. In the following section, we first provide some background on the software evolution activities before we discuss the key observations with respect to our research questions on information needs for artifacts and their dependencies in a real world context.

A. Software Evolution Activities

Participants in our study perform a variety of activities during software evolution. From the collected data, we identified a total of 403 activities from which we identified 26 unique activities. Figure 1 provides an overview of these activities and illustrates how much time on average each stakeholder per role spent on a given activity per day. The figure demonstrates that multiple stakeholder roles regularly perform the same activity. In total 19 of the 26 unique activities were performed by more than one stakeholder role. Four activities—attending meetings, project management, communication / interaction, and bug triaging—were mentioned by all five stakeholder roles. While software developers did not mention bug triaging as one of the top 5 activities during the diary study, multiple software developers mentioned bug triaging as a typical activity in their follow-up interview. Most stakeholders agree that it is a very important activity, *e.g.*,

"Bug triaging is the most important process [..] I wouldn't know what to do without this process." (P1)

Figure 1 also shows that, while a lot of time is spent each day on meetings and project management by all stakeholder roles, there are activities specific to a single or a few stakeholder roles that take up a lot of time. Department management, for instance, an activity specific to a line manager's work, consumes a big chunk of his daily activities and consists of subactivities related to running the department, such as team evaluations, vacation planning, training, promotion, resources allocation and HR acquisition. Another example is reverse engineering, in this case referring to high-level architecture recovery, which was solely performed by requirements engineers. Given the role distribution of our study participants in each team with one line manager and up to six software developers per team (see Table I), only an average of 2% of all participants' time per day was spent on bug triaging, an

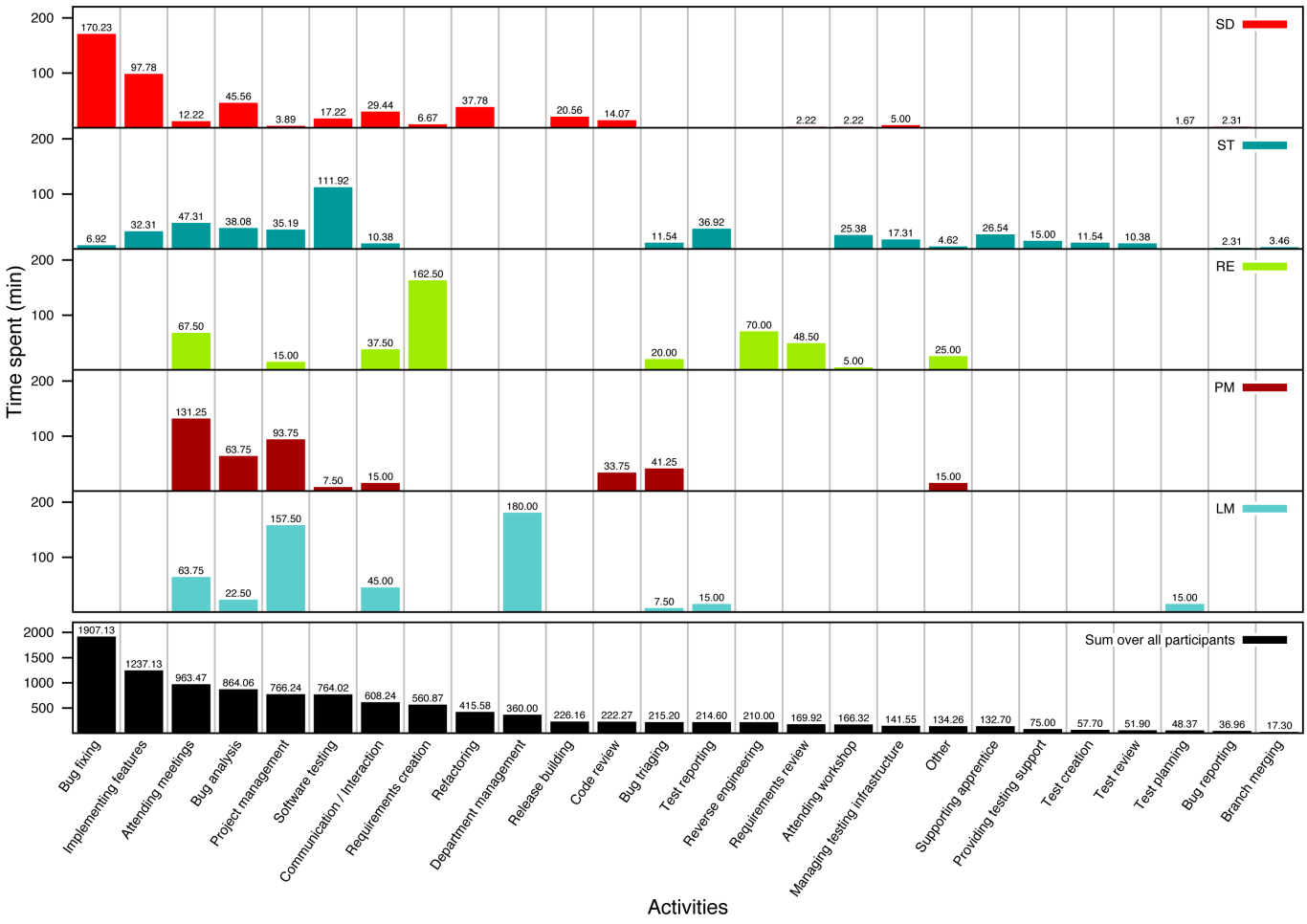

Fig. 1. Overview of the unique activities reported by the various software development stakeholders in our diary study. The height of each bar depicts the average time a stakeholder role spent on an activity per day. The last row depicts the time that all participants of both teams spent in total per day on an activity (Σ).

activity that is performed by all 5 stakeholder roles, while bug fixing consumed 18.1% and implementing features used up 11.7% of both team's time per day and was only performed by software developers and testers.

The most common activities in a software development team vary with respect to a stakeholder's role. Table II lists the three most common activities per stakeholder role with the average time that was spent on a given activity per day and participant, and the frequency with which the given activity was mentioned in the diary study. The table shows, for example, that bug analysis, an activity that is often related to software developers and testers, is among the top three activities of project managers in our study with them spending more than an hour a day on it. From the interviews, bug analysis was referred to as the activity of assessing a bug report to check if it really denotes a bug and if all important information to fix this bug is available, before it is assigned to a developer to fix it.

TABLE II
MOST COMMON ACTIVITIES PER STAKEHOLDER ROLE WITH THE AVERAGE TIME SPENT ON IT PER DAY AND PARTICIPANT AND THE TOTAL NUMBER OF TIMES THE ACTIVITY WAS REPORTED.

Stakeholder	Activity	Av. time (min)	Frequency
Software developer	Bug fixing	170.23	48
	Implementing features	97.78	22
	Bug analysis	45.56	18
Software tester	Software testing	111.92	18
	Attending meetings	47.31	12
	Bug analysis	38.08	12
Requirements engineer	Requirements creation	162.50	9
	Reverse engineering	70.00	6
	Attending meetings	67.50	10
Project manager	Attending meetings	131.25	12
	Project management	93.75	7
	Bug analysis	63.75	9
Line manager	Department management	180.00	4
	Project management	157.50	8
	Attending meetings	63.75	5

B. *Information Needs for Software Evolution Activities*

To perform their daily activities, stakeholders frequently have to seek, manage and modify large amounts of information

artifacts. These artifacts are acquired by stakeholders using a lot of different tools and accessing various repositories.

Information Needs Exhibit a Crosscutting Nature Over Information Artifact Kinds. To successfully perform a single activity, stakeholders require many different kinds of information artifacts. For instance, for bug fixing stakeholders reported to use code artifacts, change sets, planning documents, change requests, code documentation, logs, test cases, code models, configuration files and web sites. During the interviews, one software developer stated:

"[For bug fixing] I look at the CR and check if the bug is reproducible. Sometimes I use a simulator to test this. [..] Afterwards, using a debugger I can see where the problem is and then I look at the source code to see where I have to change something." (D3)

Table III presents the information artifacts that are being used for each of the 15 most common activities illustrated in Figure 1. For all activities, more than one kind of information artifact is being used. On average, 6.9 different information artifacts are being used for each of the top 15 activities shown in Figure 1 and 4.9 information artifacts are being used on average for all 26 activities we have observed during the diary study. In addition to the information artifact kinds shown in Table III, stakeholders also mentioned that they used a lot of different tools and repositories, relied on their personal experience, and also communicated a lot with other stakeholders as they performed activities. In this paper, we focus on the information artifact kinds and fragments, leaving other aspects, such as tools, experience and communication to future work.

The crosscutting information needs vary by stakeholder role for each activity. While, for example, software developers mentioned six different kinds of artifacts that they use for performing bug analysis—change requests, code, requirements, test cases, logs and code models—line managers only used change requests and requirements.

Although less crosscutting, the varying needs of different stakeholder roles can also be seen for bug triaging, an activity that all stakeholder roles perform. While all stakeholder roles reported to use change requests every time they do bug triaging, requirements engineers additionally rely on requirements specifications, code documentation and planning documents, while project managers, for instance, only occasionally use the code base but no documentation or requirements.

Over all activities, the most commonly used artifact kind are change requests. A change request can either be a bug report, a feature request or an improvement. For the top 15 activities, change requests were mentioned in 22.2% of the cases as an important information artifact for performing the given activity. During the interviews, various participants reinforced the importance of change requests by stating how significant a change request is for code-related activities:

"For each change in the code - even a single line of code - there is a change request." (D1)

"Every implementation work is based on a change request. There is no coding without a change request." (D4)

Information Needs Per Artifact Are Fragmented. To successfully perform a software evolution activity, not only do stakeholders need information from various kinds of artifacts, the information needs per artifact are fragmented and vary by activity and stakeholder role. In our study we identified this fragmentation of relevant information within information artifacts predominantly for change requests, the most commonly used artifact. Change requests contain various information such as the summary, the description, the creator, the iteration it is planned for or the resolution state. Even though all stakeholder roles use change requests when performing an activity such as bug triaging, there is a difference in the fragments that are considered important in a change request during this activity. For instance, a line manager stated that the creator of a change request is the most important information fragment while bug triaging, while a project manager claimed that the type and the severity are the most important fragments in a change request for the same activity:

"Especially important is the person who has reported the change request." (M2)

"The type of the change request is important. If it is a defect, the severity is important." (P1)

For software testing, a software tester named the resolution as the most important fragment while a software developer stated that the reproduction steps are most important:

"The resolution is one of the most important piece of information [in a change request]. It is the main mean of communication between developer and tester." (T4)

"Most important are clearly the steps to reproduce a bug. Everything else can be derived from that." (D3)

These examples for bug triaging and software testing also illustrate that the relevant information fragments of change requests vary for different activities.

As already pointed out by de Souza *et al.* [19], change requests serve stakeholders as boundary objects [28] to communicate with other team members and to align the coordination needs. A boundary object is generally defined as an artifact that is able to support the communication between different groups by providing a common content that can be accessed and interpreted by all groups [29]. As mentioned above, change requests are the most common and central information kind in all software evolution activities of the two teams we studied. Participants used them to steer the workflow and to pass information to other stakeholder roles. For each stakeholder role though, different fragments of the change request are considered important. For instance, one participant stated that after fixing a bug, the responsible developer enhances the information in the change requests and adds a description of his resolution steps. For testing the bug fix, the software tester then reads these resolution

TABLE III
INFORMATION ARTIFACT KINDS USED FOR EACH OF THE 15 MOST COMMON ACTIVITIES.

Activity	Code	Requirements	Release notes	Change set	Planning documents	Change request	Code documentation	Test report	Personal notes	Logs	HR-related documents	Test case	Code review	Code model	Configuration file	Website	# of artifacts
Bug fixing	■			■	■	■	■			■		■		■	■	■	10
Implementing features	■	■		■	■	■	■		■			■	■	■		■	11
Attending meetings		■			■	■	■	■	■								6
Bug analysis	■	■			■	■	■	■		■		■		■	■		10
Project management	■	■			■	■	■			■	■	■					8
Software testing	■		■			■	■	■				■		■	■		8
Communication/Interaction	■	■				■	■	■	■	■		■		■	■	■	11
Requirements creation	■	■				■	■		■						■	■	7
Refactoring	■						■					■			■		4
Department management		■									■						2
Release building	■		■			■	■		■					■			6
Code review	■	■		■			■						■				5
Bug triaging	■	■			■	■	■										5
Test reporting		■				■	■	■		■		■					6
Reverse engineering	■	■					■								■		4

steps to understand what to test and how to adapt test cases if necessary.

Crosscutting and Fragmented Information Needs Vary by Activity and Role. The collected data shows that many information needs in software development exhibit a crosscutting nature. They cut across stakeholder roles, activities, artifacts and even fragments of artifacts. For each activity performed during software development, a multitude of different information artifacts is being used. While different stakeholder roles perform some of the same activities on a daily basis, different roles use different information artifacts to perform the same activity. The crosscutting nature of the information needs is even visible for a single artifact, since different roles use different fragments of the same artifact for the same activity. This is particularly evident for change requests, the most commonly used artifact in the observed software development teams, for which certain fields are only relevant to certain stakeholder roles.

C. Information Artifact Inter-Dependencies

A lot of dependencies exist between artifacts, in particular in the development and evolution of software. There are dependencies between a requirement and its implementation in the code or the test cases that cover the requirement, structural relations between different code artifacts and the link between a change request and the change sets to resolve it, to name just a few. To successfully perform the daily activities, these relations are an important part of stakeholder's information needs to identify and understand the relevant fragments of artifacts. Research often talks about these inter-dependencies in terms of traceability links that are useful for instance for requirements validation, impact analysis and program comprehension (*e.g.*, [6], [5]).

Inter-Dependencies Relevance Varies by Activity and Stakeholder Role. Participants in our interview study talked a lot about the dependencies between artifacts and the communication efforts to identify them. In particular, the dependencies between requirements or versions of a requirement and code or change requests were mentioned continuously:

"Developers most frequently ask where to find the specifications for a specific component." (R1)

"If there is no requirements document attached to the change request, it is often necessary to ask a requirements engineer to get the appropriate document." (D6)

Overall, interdependencies were mentioned as an important part to successfully perform software evolution activities with a lot of time being spend on recovering them.

Similarly to the crosscutting nature of information needs, the relevance of dependencies between artifacts also changes with stakeholder role and activity. For the same activity, different stakeholders reported different inter-dependencies as most relevant. For instance, for the analysis of a bug reported by a field worker, a line manager (M2) stated that he first examines dependencies between the reported bug and requirements by talking to the requirements engineers and investigating whether it is a bug or a feature. A software tester (T5) said that the reproduction of the bug is most important

and he therefore first examines which component is affected and then talks to the software developer who wrote the code. Finally, a software developer (D3) mentioned that he first analyses the configuration file that was in use when the bug occurred to see if there is a problem in there before examining the code.

In other cases, the same interdependency is required for different activities. Participants in our study repeatedly talked about the links between a code fragment and the responsible software developer for various activities. While a project manager (P1) stated to require this link in the process of bug triaging, a software tester (T5) said he uses the link for bug analysis to find out more information on how to test the bug, and a software developer (D4) stated to use this link when he reviews code using a static analysis tool and wants to talk to the responsible developer about the problems he found.

Links between Information Artifacts are Often Missing. While some information artifacts can be explicitly linked, such as test cases or requirements to change requests, we observed that these links are often out of date, unreachable or not available at all:

"Requirements documents are difficult to find. There are a lot of different places to store a requirements document. It is not always linked to a change request and it is not even always clear if there is a requirements document and if there is, where it can be found." (D8)

"I guess that in only 6% of all cases there is really a link between the test case and the requirements." (R2)

In the interviews, 14 participants (70%) explicitly mentioned the lack of links between artifacts in their daily activities and the problems the missing links cause.

Links are Missing for Many Reasons. Participants mentioned a variety of reasons for the lack of explicit links. Several stated that there are a lot of different systems and repositories and it is not always clear where to best store or find artifacts and links, *e.g.*,

"There is a lot of documentation available, but it is not widely known where to find it." (D9)

The rapid evolution and high frequency of change in the artifacts were also mentioned to make it difficult for keeping links up to date, *e.g.*,

"Developers do not know which requirements documents they have to update if they change something in the code, because they can't find the associated requirements documents." (P1)

In addition, a lot of the software systems that participants are working on are legacy systems. When they first started working on these systems, requirements specifications, documentation, as well as other information artifacts were not fully available and the main focus was on the development of the code. Therefore, links between various information artifacts were often not established. Recovering and establishing these links today is not a priority, in particular since it requires a lot of time and effort while the systems continue to rapidly evolve.

Missing Links Lead to Additional and Repeated Communication Effort. The missing links between information artifacts and fragments lead to several problems. Multiple stakeholders have problems to identify the dependent information artifacts or fragments themselves and therefore communicate a lot with other stakeholders. Participants mentioned the time-consuming communication effort in particular for requirements:

"If there is no requirements document attached to the change request, it is often necessary to ask a requirements engineer to get the appropriate document." (D6)

"4-5 iterations [with a requirements engineer] are necessary until every issue is clarified and until I can start to implement anything." (D2)

Since links between artifacts are not captured explicitly, participants also have to repeatedly explain the same interdependencies to other stakeholders:

"The same clarification requests have to be answered again and again." (R3)

"I have to ask the requirements engineer again and again for clarification [...] Some requirements documents have dependencies on other documents. They overwrite information in other documents. It is very difficult to find the appropriate and complete description." (D2)

In our data analysis, we observed that almost all stakeholder roles spend a lot of time on link retrieval. These observations also explain why there is so much communication and interaction reported from various stakeholder roles in Figure 1.

Wikis are Used to Compensate for the Missing Links. To overcome the often time-consuming interaction with other stakeholders, participants started to use wikis. In the organization we observed, people started using wikis mostly a few months and up to a year before our interviews took place, even though the projects have been existing for several years. In general, the wiki provides a single-entry point to the participants where a lot of the missing links are kept:

"The wiki particularly contains links to already existing documents. It provides a single-entry point for a lot of documentation and information. For example I add a link to a software release. Then the testers use this link to get the newest software versions for their simulators." (D7)

"In the wiki links to other important internal documents, e.g. requirements specifications, are stored." (D3)

Wikis are used to manage a lot of different information artifacts that are used in a lot of different activities. However, these collection of links and pointers to information artifacts and fragments are not very structured. The wiki is largely

a collection of information, such as instructions on how to install or use tools, instructions on how to set up projects and environments, release notes for each software version linking to the list of change requests for each release and the known bugs fixed in the release, as well as outstanding bugs, and links to technical documentation.

During the follow-up interviews, we observed that 11 out of 20 interviewed stakeholders use the wiki on a regular basis. We also observed that not all stakeholders think that the wiki is a good solution for storing the links between various information artifacts, *e.g.*,

"We in the requirements engineering team do know that developers use a wiki, but we do not think that this is useful." (R2)

V. Discussion

The variety and crosscutting nature of information needs in software evolution activities in combination with the large amounts of continuously changing software project information, make it challenging for stakeholders to satisfy their information needs in a timely manner. In our study in the workspace of stakeholders, we observed several participants jumping back and forth between a multitude of tools, editors and repositories for a single activity, since each one of them only presented one kind of artifact. In addition, participants reported on spending a lot of time on communicating with other stakeholders to identify dependencies between artifacts as well as discussing the artifacts.

Need for Aggregating Information Artifacts. The need for stakeholders to switch between tools and gather a variety of crosscutting information fragments to perform a single activity, creates cognitive burden and requires time and effort. Rather than providing one view per artifact, tool support is needed to aggregate and synthesize the multitude of artifacts that stakeholders work with these days. This was also partially mentioned by participants:

"There is a need to unify all the internal repositories, documents, processes, etc." (M2)

"The main problem is that there is no search function that covers all the databases." (R3)

However, we hypothesize that it is not only important to provide access to multiple kinds of artifacts as asked for by the participants, but that new techniques are provided that aggregate various kinds of artifacts in a single view or presentation.

Tailoring Views Based on Role and Activity Context. In our study we found that the information fragments used for a given activity vary depending on the activity and stakeholder role context. By providing support for tailoring the aggregated information to the activity and role, we might be able to cut down large portions of information, such as irrelevant fields in a change request. This in turn would allow us to focus stakeholders' attention to artifacts that are really relevant for their current activity.

Lack of Socio-Technical Congruence. The high communication load between stakeholders for their information needs in combination with the missing links between artifacts points to a lack of socio-technical congruence. Socio-technical congruence refers to the idea of "fit" between the social and the technical dimension in software development [30], [31]. It has previously been shown that teams are more efficient, when the technical links and the communication structure is congruent [30], [32].

One approach to overcome the problem of missing links is the usage of wikis to manage an unstructured collection of links. Other companies might not use a wiki but other techniques, strategies and tools to try to overcome the missing linkage between information artifacts, such as dashboards [33], or explicit iteration plans.

We also observed participants pointing out that the problems of missing links are increasing with the increasing size and distribution of teams:

"The requirements engineer no longer hears what we developers are talking about and therefore he can no longer intervene if the developers plan to implement something in a way that is not correct from a requirements perspective." (D8)

Making Links First Class Entities. Current repositories and tools that maintain information artifacts store links between information artifacts most often as by-products of the information artifact itself. Furthermore, there are often multiple places in these artifact repositories to store these links. This leads to links being neglected, not updated or not available and makes it difficult for stakeholders to identify and find the relevant links.

We suggest to make artifact links first class entities so that it is easier to search for, find and update links between artifacts. We hypothesize that this will result in more explicit links and in turn a higher socio-technical congruence and a reduced communication effort between stakeholders. Since humans are good at recalling associations [34], links as first class entities can also allow stakeholders to more efficiently find relevant information by querying for associations rather than just artifacts, as related research has already shown [35].

VI. Conclusion

In this paper, we presented the results of an empirical study investigating stakeholders' information needs for software evolution activities. The study consisted of a six day diary study and a follow-up interview and was conducted with 23 stakeholders of two professional software development teams. The focus of our study was on providing evidence for the information needs of multiple stakeholder roles and how they manifest themselves in the context of a real world company.

From the analysis of the collected quantitative and qualitative data, we found that information needs exhibit a cross-

cutting and fragmented nature. Thus, to successfully perform their daily activities, stakeholders require multiple different kinds of artifacts or fragments of these artifacts, and these required information fragments vary by stakeholder role and activity. Furthermore, we observed a lack of socio-technical congruence: dependencies between information artifacts are often not explicitly captured or out of date and and require stakeholders to repeatedly put additional effort into communicating with other stakeholders to understand and recover these missing links.

We suggest that approaches to support multiple stakeholder roles with their information needs should provide means to aggregate multiple kinds of artifacts in a single view that can be tailored by stakeholder role and activity and that dependencies between information artifacts should be made first class entities. Future work will look into extending our study to further teams and companies and examine generalizability, as well as we plan on developing concrete techniques to support the crosscutting information needs as suggested.

REFERENCES

[1] D. Perry, N. Staudenmayer, and L. Votta, "People, organizations, and process improvement," *IEEE Software*, vol. 11, pp. 36 –45, July 1994.

[2] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, pp. 69–81, 1995.

[3] D. Čubranić, G. C. Murphy, J. Singer, and K. S. Booth, "Hipikat: A project memory for software development," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 446–465, 2005.

[4] A. Begel, Y. P. Khoo, and T. Zimmermann, "Codebook: discovering and exploiting relationships in software repositories," in *Proc. of the 32nd ICSE '10*, pp. 125–134, ACM, 2010.

[5] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 970–983, 2002.

[6] J. Cleland-Huang, C. K. Chang, and M. Christensen, "Event-based traceability for managing evolutionary change," *IEEE Transactions on Software Engineering*, vol. 29, no. 9, pp. 796–810, 2003.

[7] W. L. Johnson and A. Erdem, "Interactive explanation of software systems," in *Proc. of the 10th KBSE '05*, pp. 155–164, IEEE Computer Society, 1995.

[8] J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories," in *Proceedings of the 31st ICSE '09*, pp. 298–308, IEEE Computer Society, 2009.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: improving cooperation between developers and users," in *Proceedings of the CSCW '10*, (New York, NY, USA), pp. 301–310, ACM, 2010.

[10] J. Sillito, G. C. Murphy, and K. De Volder, "Asking and answering questions during a programming change task," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 434–451, 2008.

[11] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proc. of the 29th ICSE '07*, pp. 344–353, IEEE Computer Society, 2007.

[12] T. D. LaToza and B. A. Myers, "Developers ask reachability questions," in *Proc. of the 32nd ICSE '10*, pp. 185–194, ACM, 2010.

[13] T. Fritz and G. C. Murphy, "Using information fragments to answer the questions developers ask," in *Proc. of the 32nd ICSE '10*, pp. 175–184, ACM, 2010.

[14] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?," in *Proc. of the ICSE' 12*, pp. 255–265, IEEE Press, 2012.

[15] C. Seaman, "The information gathering strategies of software maintainers," in *Proc of the ICSM '02*, pp. 141–149, 2002.

[16] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: a study of developer work habits," in *Proc. of the 28th ICSE '06*, pp. 492–501, ACM, 2006.

[17] J. Singer, T. Lethbridge, N. Vinson, and N. Anquetil, "An examination of software engineering work practices," in *CASCON First Decade High Impact Papers*, CASCON '10, pp. 174–188, IBM Corporation, 2010.

[18] A. Schröter, J. Aranda, D. Damian, and I. Kwan, "To talk or not to talk: factors that influence communication around changesets," in *Proc. of the ACM CSCW '12*, pp. 1317–1326, ACM, 2012.

[19] C. R. B. d. Souza, D. Redmiles, G. Mark, J. Penix, and M. Sierhuis, "Management of interdependencies in collaborative software development," in *Proc. of the ISESE '03*, pp. 294–303, IEEE Computer Society, 2003.

[20] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?," in *Proc. of the 33rd ICSE '11*, pp. 804–807, ACM, 2011.

[21] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proc. of the 25th ICSE '03*, pp. 125–135, IEEE Computer Society, 2003.

[22] X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," in *Proc. of the 26th IEEE/ACM ASE '11*, (Washington, DC, USA), pp. 223–232, IEEE Computer Society, 2011.

[23] S. Acuna, N. Juristo, and A. Moreno, "Emphasizing human capabilities in software development," *IEEE Software*, vol. 23, no. 2, pp. 94 – 101, 2006.

[24] M. Yilmaz, R. O'Connor, and P. Clarke, "A systematic approach to the comparison of roles in the software development processes," in *Proc. of the 12th SPICE '12*, pp. 198–209, Springer Berlin Heidelberg, 2012.

[25] D. Rodríguez, M. A. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[26] P. C. Pendharkar and J. A. Rodger, "An empirical study of the impact of team size on software development effort," *Information Technology and Management*, vol. 8, no. 4, pp. 253–262, 2007.

[27] P. Y. Martin and B. A. Turner, "Grounded theory and organizational research," *The Journal of Applied Behavioral Science*, 1986.

[28] S. L. Star and J. R. Griesemer, "Institutional ecology, 'translations' and boundary objects: amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39," *Social Studies of Science*, vol. 19, pp. 387–420, 1989.

[29] C. Kimble, C. Grenier, and K. Goglio-Primard, "Innovation and knowledge sharing across professional boundaries: Political interplay between boundary objects and brokers," *International Journal of Information Management*, vol. 30, no. 5, pp. 437 – 444, 2010.

[30] M. Cataldo, J. D. Herbsleb, and K. M. Carley, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity," in *Proc. of the 2nd ACM-IEEE ESEM '08*, pp. 2–11, ACM, 2008.

[31] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley, "Identification of coordination requirements: implications for the design of collaboration and awareness tools," in *Proc. of the 20th Anniversary CSCW '06*, pp. 353–362, ACM, 2006.

[32] F. Bolici, J. Howison, and K. Crowston, "Coordination without discussion? socio-technical congruence and stigmergy in free and open source software projects," in *2nd International Workshop on Socio-Technical Congruence, ICSE '09*, 2009.

[33] C. Treude and M.-A. Storey, "Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds," in *Proc. of the 32nd ICSE '10*, pp. 365–374, ACM, 2010.

[34] E. Tulving and D. M. Thomson, "Encoding specificity and retrieval processes in episodic memory," *Psychological Review*, vol. Vol. 80, no. 5, pp. 359–380, 1973.

[35] D. H. Chau, B. Myers, and A. Faulring, "What to do when search fails: finding information by association," in *Proc. of the SIGCHI '08*, pp. 999–1008, ACM, 2008.